An Approach of Reuse-based Software Process Improvement

Ruzhi XU1, Peiguang LIN1,†, Zhikun ZHAO1, Leqiu QIAN2

1 School of Computer & Information Engineering, Shandong University of Finance, Jinan 250014, China
2 Department of Computer Science and Engineering, Fudan University, Shanghai 200433, China

Abstract

This paper puts forward a novel approach of applying process reuse technology to implement software process improvement and control. A united framework of reuse-based software process improvement is proposed, which integrates the model-driven process improvement (Top-down) with measurement-driven process improvement (Bottom-up). The method of component-based software process instantiation and adaptation are presented. A new process component retrieval model based on tree-inclusion matching is given, which can greatly improve the recall while maintaining a high level precision.

Keywords: Process Reuse; Process Improvement; CMMI

1. Introduction and Related Works

Process improvement is a comprehensive and continuous activity, which involves not only every basic activity during the process modeling, process implementation, but also involves the process measurement, process assessment, process optimization and control [1,5,6]. Among them, the approach of process improvement determines the relevant technologies of implementing the process improvement.

Currently, there are mainly two kinds of modes to implement process improvement, one is the model-driven, and the other is measurement-driven [2,5]. The former, ISO 9000[3], CMMI/CMM (Capability Maturity Model)[4,7] for instance, aims at improving organization’s process ability maturity, implements Top-Down measurement, and launches relevant improvement activities based on a definite assessment model. The latter constantly collects feedback from the process measurement activities, and takes improvement actions to solve the problems produced in the process execution [7,8].

Model-driven process improvement (Top-Down), can control the improvement activity within the scheduled range, and can produce the satisfied improvement result consuming as little resources as possible, because it has clear improvement goal. This kind of process improvement focuses on setting up and improving the process model [4,6], which reflects the commonality of a generality of software projects in an organization. Therefore the improvement mode is more effective than that for a concrete project. But, this kind of process improvement mode is too universal to involve the specific knowledge and technology

† Corresponding author. Peiguang Lin

Email addresses: llpwgh@sdfi.edu.cn (Peiguang LIN).
of the projects. In addition, as it is always based on certain assumption, the risk will often be brought on because of defective planning of improvement goal and model, which even makes the previous efforts futile.

On the opposite, measurement-driven process improvement mode (Bottom-Up) concerns about the improvement of specific project process and single software product. The improvement goal of the process is unique and closely relative to a specific domain. The improvement driving force comes from remedy various kinds of process defects and deficiencies found through the measurement. The process improvement, which directs against concrete defects, is often effective, and it can guarantee every improvement activity to get a more optimized process. However, such bottom-up process improvement mode lacks of complete planning and outstanding theme, which is unfavorable to the forming and accumulation of process knowledge, makes the process improvement efficiency very low [2,7,9].

Obviously, it is an ideal process improvement mode to combine Top-Down together with Bottom-Up above-mentioned, which can not only promote the organization’s process maturity, but also guarantee the success of concrete projects.

At present, process reuse becomes an important field in software engineering related to the reuse of the information produced during previous software development projects, in order to decrease the effort needed for a new project [1,6]. Process reuse defines a wide area of research and practice related to different aspects of the reuse of knowledge obtained from previous successful projects[10,12,13]. Many of the questions under this topic are related to the reuse of process assets. Other approaches [14,15] describe procedures to define generic process (or process templates), which are abstract models to be reused as the starting point for the modeling of processes in similar contexts. From a generic process, a user can adapt generic process descriptions to specific domains [9]. At CMM level3 or higher [4], the organization standard software process (OSSP) is repeatedly reused under different (but similar) contexts, to provide guidelines and support for organization’s projects.

This paper puts forward a novel approach of applying process reuse technology to implement software process improvement and control, presents an united framework of reuse-based software process improvement, which integrates the model-driven process improvement (Top-down) with measurement-driven process improvement (Bottom-up), discusses the method of component-based software process definition, process instantiation, dynamic adaptation and process retrieval.

2. Reuse-based Process Improvement Framework

Reuse-based software process improvement framework is a united framework which introduces measurement-driven process improvement into model-driven process improvement. Project-level process improvement reuses organizational process assets. The data and process documents produced in the project execution process are validated, abstracted, packed, and then stored in the organizational process assets library shared in the whole organization. Through regular statistical analysis to the operation results of the process, the process baseline is upgraded constantly, new opportunities for process improvement is found to launch a new-round organization-level process improvement. Circulating in this way, the process can be improved continuously both in organization and project level.

Fig.1 shows a process framework to implement reuse-based software process improvement and control. In the framework, process assets are reusable artifact, process document, process metrics data, which are accumulated in the improvement progress. It includes the software life cycle model, organizational standard software process (OSSP), process tailoring direction, process components (PCs), process data, and software process-related documentations and so on.
Organization-level systemic circulation regards organization’s standard software process (OSSP) as the target of major improvement. It is divided into four stages, process definition, process implementation, process assessment and process improvement. All of these stages integrate into a circulation cycle to improve the ability maturity of software organization constantly. Through the implementation and process studies of a lot of software projects, a large amount of process data and other process assets have accumulated in the organization process assets library, some defects are found, and new process improvement goal is made. The feedback will push process model and organization process assets evolved, thus constantly optimize the reusable software process, and enter a new round process operations based on new-defined process. Each circulation makes the organization’s software process more stable, controllable and predictable, and accumulates more reusable software process assets.

Project-level micro-circulation regards software process of the project (process instance) as the target for improvement. It is made up of four stages, process instantiation, process operation, process measurement, adaptation and optimization. All of these stages integrate into a circulation cycle to dynamically adjust and optimize the software project process, to assure the achievements of the project target. To the concrete project, through tailoring and exampling the organization OSSP, the executable software process forms. Then in the process operations, it constantly optimizes the project process until the project is completed by ways of exampling and adjusting dynamically the software project process through process measurement, tracking and optimization according to project control requirements and environment change. The reusable process assets and process information in the organization assets library offer reliable foundation for realizing the above process instantiations and dynamic adaptations. At the same time, if the measurement of the data and relevant operation process is validated, packed, and then store them in process database as the resources to form organization reusable process assets.

It realizes dynamic process adaptations and optimizing control to assure the achievement of project goal.
using process components and relevant process information in the organization assets library according to the project environment change in the operational process. It also provides process assets for organizations, and validates the process improvement target given by process macro-circulation. The establishment of the process assets library makes it possible to accumulate and enrich process improvement knowledge, experiences, and other process assets continuously. It will promote the process maturity, lay foundation for dynamic tracking and control of the process, and establish a constant improving process depending on quantity control.

3. Reuse-based Process Dynamic Adaptation

A defined process can not be adapted to all projects and all situations. Reuse-based software process instantiation goes on the evolving way to combine the initial static instantiation with dynamic adaptation. Because the static instantiation at initial static instantiation is the foundation of the project implementation, now we provide the method of static instantiation of reuse-based software process at first, then discuss how to dynamically adjust countermeasure based on reusable software process in several situations of process change.

3.1. Initial Instantiation for Reuse-based Software Process

To the software process of a concrete project, the static instantiation of the process begins at the planning stage. According to the characteristic of the specific software project, it utilizes process reuse guide to tailor properly the general description of the process model at first, and forms the project defined software process (PDSP), and then evaluates to every basic composition and parameters of interaction of PDSP.

Fig.2 shows the activity flow of reuse-based software process instantiation, which describes the instantiation procedure of basic composition of software process. The instantiation content includes activity instantiation, role instantiation, product instantiation, resource instantiation etc. The instantiation method starts with activity instantiation; following are the instantiations of other basic items accordingly. Among the mentioned basic items, activity name, role name, activity state, product name, necessary resources, following activity set, synchronous activity set, activity starting time and finishing time should be initialized. The process database lays data foundation for process instantiation. At the planning stage of the project, users Requirement has been already clear. The project manager finds out reusable process components or relevant data information from organization process database, according to the characteristic of the project, such as the application of the project, suitable technology, method, goal and output finished. The tactic is as follows:

- First of all, search for process components suitable to the current organization process definition from process component library. All of the components should be same or similar to present process. Then the components can be used after adjusted properly according to the need;
- If there is no process component or data with higher degree of similarity, we should search baseline data of project process in the same domain, then adopt the average of them;
- If there is no similar project in the process database and there is no project process baseline data in the same domain, organization process ability baseline data in common use should be adopted.

The above-mentioned process data should be adjusted properly according to the concrete or special
factors of the project. These factors include the mutability of the predictable demand, the clarity of the demand, the customer's participation desire and ability, etc.

3.2. Process Adaptation

After initial instantiation, a software process becomes an executable project plan. As the actual software process is more complex, and long-time process cycle will make it have more uncertainty both in the structure and attribute too, dynamic adaptation to the software process must be carried on according to the actual situation of the project during the execution of the process, such as revising the schedule plan and redistributing resources, etc.

Summarizing the actual process change during its execution, we conclude that dynamic changes of the process can be classified into the following four kinds of basic types (shown as Fig.3):

Accordingly, we put forward four kinds of adjustment tactics based on process reuse.

(1) Trivial

In this case, the structure of the process or the dependency relations between the activities has no change, but the attribute state of the process has changed. For example, the sequence and content of a project’s activities such as system requirement analysis, design, code and system testing have no change, but participants and the time of beginning and ending of the activity can be different. If any activity was delayed in the process, the following activity’s attribute values must be adjusted to remedy the time delay so that the project can be delivered on schedule.
The dynamic adaptation countermeasures adopted: To adjust the state attribute parameter of the process only, as the structure of the process has not changed, the following process components can be instantiated again based on the historical data/information of the similar process in the process database. This kind of dynamic adjustment is the most common reuse case and very easy to be realized in practice.

(2) Limited.
In this case, both the attribute state and the activity of the process have changed, but the changes are confined within a sub-process or a process component, and the macroscopic structure of the whole project process has no change generally. For example, the testing activity in a software process is scheduled to be executed two times, but it failed to reach the expectation in practice, so the third time is needed.

The adjustment countermeasure based on reuse is: To look for the same or the similar process component to the new process component structure and implementation from the process component library. Otherwise, it is needed to add some activities from the original process component, and be instantiated the using the process data acquired from the process database again after adaptation.

(3) Feedback.
In this case, though the structure of the process has changed during its execution, but the software process is still the processes that has been already defined in advance, so only the process components whose structure have changed need to be adjusted, while other process components’ attribute value is adjusted only. For example, some mistakes belonging to the detail design are found in the phase of unit test, the information are documented and transferred to the group of the detail design. After the modification of the mistakes, the process goes on.

This situation can be regarded as the composition of the two situations above-mentioned.

(4) Overall.
In this case, attribute of the process and topological structure have changed fundamentally both, the process being carried out after is the software process defined in advance. New software process is needed to be selected or defined again and then be instantiated. For example, a requirement mistake is found while the project is handing over, the problem is documented and transferred to analysis group to re-analyze it, but the following work are wrapped and outsourced to complete design, code and testing rather than following the original process to carry on the design, coding, integration and testing.

The adjustment countermeasure based on reuse is: To set up the new software process based on process components and historical information in the process database. To look for the same or the similar process components from the process assets library and reuse them to construct a new one. Otherwise, it is needed to add some activities from the original process component, and be instantiated the using the process data acquired from the process database again after adaptation.

All the information above-mentioned including the scenario and the measures that have taken must be documented and stored in the process database for the future reuse in the similar situations.

4. Process Component Retrieval
The key of the process reuse is to find candidate process components for reuse. The first step to retrieve a process component is the formulation of a component query. The more detailed the query, the higher the
possibility of finding a detailed reusable process component. Nevertheless, even a vague description of the future project may lead to the proposal of a reuse candidate.

As XML offers a uniform and standardized way to represent and exchange both documents in the sense of information retrieval and data exported from databases, and the document type definition function of XML endues a level intension toward process modeling and process component description, which matches the process structure. As a result, xml has been widely used as the process modeling and process component description metal language.

XML-based process component retrieval can be changed into matching problem between component query tree of and component description trees stored in the component library [12,16]. It is a similarity search and ranking. The smaller the matching cost between two trees is, the better the match between two trees is. In this paper we present an approach to find approximate answers to formal user queries. We reduce the retrieval problem of answering process component queries against XML document collections to the unordered Tree containment problem.

Compare with the definition of traditional Tree-embedding and Tree-inclusion [17], the Tree-containment requires less restricted but more relax matching conditions. Following, we give the Tree-containment definition and the query matching cost based on tree-containment first.

**Definition 1.** (Tree-containment). Suppose \( Q = (V,E,\text{root}(Q)) \), \( D = (W,F,\text{root}(D)) \) is two unordered labels trees which is named process component query tree and process component description tree respectively, \( V' \subseteq V, W' \subseteq W \). If exists a mapping \( V' \rightarrow W' \), makes to all \( u, v \in V' \) meets the following requirements.

1) \( u = v \Leftrightarrow f(u) = f(v) \ u, v \in \text{Domain}(f) \);
2) \( \text{label}(u) \approx \text{label}(f(u)) \);
3) \( u = \text{ancestor}(v) \Leftrightarrow f(u) = \text{ancestor}(f(v)) \).

We call \( f \) is a Tree-containment matching from \( Q \) to \( D \), abbreviated as the Tree-containment matching.

**Definition 2.** (Matching cost) Suppose \( Q = (V,E,\text{root}(Q)), D = (W,F,\text{root}(D)) \) are two unordered labels trees, matching cost \( \text{TCostM}(Q,D) \) from \( Q \) to \( D \) is as follows :

\[
\text{TCostM}(Q,D) = \min\{\gamma(f) \mid f : Q \Rightarrow D\}
\]

Where,

\[
\gamma(f) = \sum_{v \in \text{domain}(f)} \gamma(\text{label}(v) \rightarrow \text{label}(f(v))) + \sum_{v \in V' - \text{domain}(f)} \gamma(\text{label}(v) \rightarrow \lambda) + \sum_{w \in \text{spectrum}(f) - \text{Range}(f)} \gamma(\lambda \rightarrow \text{label}(w))
\]

\[
\text{spectrum}(f) = \text{Range}(f) \cup \left \{ w \notin \text{Range}(f), \exists w_1, w_2 \in \text{Range}(f) \text{ such that } w_1 = \text{ancestor}(w) \land w_2 = \text{descendant}(w) \right \}
\]

Tree-containment allow the definition domain of \( f \) to be a subset of the query node set, supervised to the restriction on definition domain of \( f \) with the tree, so that it allow to exist surplus query information, this has lay a foundation for promoting precision of the component.
Tree containment only demands to keep nodal ancestor-descendant relation (the condition 3 of defining 1), it permits the information to be participated in matching at different layers, to improve component precision. Because this kind of match still keeps a certain ancestor's descent relation at the same time, so it still keeps the match on the semantic structure level, which differs from the simple key word query, and guarantee the component precision maintains a higher level.

The one-way relation "\( \Rightarrow \)" (condition 3 of definition 1), can reduce the time complexity of algorithm greatly under the prerequisite of not destroy the component retrieval meaning.

Tree containment only requires the label has a similar or approximate relations (condition 2 of definition 1). This improvement can strengthen the fuzzy matching ability to improve component retrieval recall.

Dynamic programming algorithm are designed for calculating the matching cost [17], which is based on Tree containment, and the results are indicated that this model can greatly improve the component recall while maintaining a high level precision.

5. Application

The approach of Reuse-based Software Process Improvement was applied in large software company (ssc). The statistical operation data for more than one year (figure 4) illustrated that the company's ability of schedule control increased by an average of 30%. While the cost of the project in the first half year increased, the average statistics cost dropped by about 15% for the whole year.

These results indicated that the new approach in this paper can not only improve the software process and increase the capability maturity of the organization, but also enhance the software project control. As a result, It can effectively promote the resource allocation of software project resources dynamically, increase the usage effect of project resources, and reduce the costs.

6. Conclusion and Future Work

This paper puts forward a novel approach of applying process reuse technology to implement software process improvement and control, presents an united framework of reuse-based software process improvement, which integrates the model-driven process improvement (Top-down) with measurement-driven process improvement (Bottom-up), and discusses the method of component-based software process definition, process instantiation, dynamic adaptation and process retrieval.
A retrieval model based on tree-inclusion matching is given, which can greatly improve the recall while maintaining a high level precision. Finally, a reuse-based process and project management system (P2MS) is introduced in this paper.

The new approach can guarantees the process improvement objectives both in the organization level and in project level at the same time. Also it can help to makes the knowledge about process improvement, experience and other process assets to be accumulated constantly and shared within the organization.

Our further research work will focus on how to apply data mining technology to the Process Database management system, excavate out valuable knowledge useful for the process improvement and project management from a large amount of process data, and transfer from general process management and process information to information management and knowledge sharing.

Acknowledgement

This work is supported by the Provincial Natural Science Foundation under Grant No.2006BS01007 and ZR2009GM030; the Project of Shandong Province Higher Educational Science and Technology Program, under Grant No. J09LG05; the Graduate Innovation Education Research Project of Shandong Province, China, under Grant No. SDYY09088.

References
